Automatic formatting of music sheets through MILLA rule-based language and engine

Automatic Formatting of Music Sheets through MILLA Rule-Based Language and Engine

P. Bellini, I. Bruno, P. Nesi

Department of Systems and Informatics University of Florence
V. S. Marta, 3, 50139 Florence, Italy
Tel: +39 055 4796523, fax: +39 055 4796363
nesi@dsi.unifi.it, http://www.dsi.unifi.it/~nesi/

ABSTRACT

Automated music formatting helps composers and copyists to speed up the process of music score editing by facilitating complex evaluations needed to write correct music sheets in terms of symbol positioning, justification, etc. Delegating this activity to the computer allows the composer to increase the efficiency by letting him/her concentrate on more creative aspects. Music formatting is one of the most relevant problems of programs for music notation editing. Several written and non-written rules are typically applied by music engravers and composers. In this paper, an approach for formalizing formatting rules and the related tools are presented. The approach is based on insertion, positioning and priority rules and on the associated conditions for their activation. Conditions and rules formalized in the proposed approach are automatically executed by a formatting engine called MILLA (Music Intelligence Formatting Language). The Open Source and free distributed WEDELMUSIC editor (www.wedelmusic.org) is based on MILLA. The approach is presented together with several examples.

Keywords: Music sheet, music formatting, music notation rules, music context, symbol insertion, symbol positioning.

1 INTRODUCTION

The music formatting process must address a large number of cases. The decision process of inserting and positioning the music notation symbols in the music sheets must cope with a huge number of variables which characterises the music score context. The main problem of standard western music formatting has to do with the adoption of a large number of rules and exceptions to produce correct music sheets (Byrd 1984). Rules and guidelines were established by copyists and music engravers during centuries and informal reference to such guidelines can be found in several books. Relevant studies and experience on music notation formatting are presented in (Roush, 1988), (Wolff, 1977), (Heussenstamm, 1987) and (Wood, 1989) and yet they do not consider aspects of music notation modeling in computer science.

Music publisher's copyists and most composers use music notation programs to produce music sheets in digital form and mainly for printing them. These programs describe the music notation by means of a symbolic description. In order to make easier the music composition, composers and music copyists expect music notation programs to provide both music formatting automation and format flexibility so as to cope with exceptions. Automation in the music formatting process helps composers and copyists to speed up the writing process of music score composition by facilitating complex evaluations needed to write correct music sheets in terms of symbol positioning, justification, line breaking, etc.

The problems that an automatic system for music formatting has to face with during the insertion and positioning of symbols in a music score have been outlined in (Byrd, 1988), (Gourlay, 1986), (Gourlay, 1986), (Parrish, 1987), (Bellini, 2001). In the above-mentioned reports, several aspects and exceptions of music notation have been considered and commented. The main problems of automatic formatting are the mechanisms leading to the insertion and positioning of symbols in the music sheets. Main rules deal with music notation symbol positioning, with respect to the other symbols being part of the music sheet. An automatic formatting engine has to evaluate parameters such as the stem direction of the note (upward or downward) and the stem length, etc., according to the music context where the note is inserted. The parameter evaluation has to take into account the presence of polyphony, if the note belongs to a chord or to a beamed

group, etc. The slope of beams depends on the note heights belonging to the beam and it impacts on the notes stem length. The positioning of markers such as accents, ornaments, expression, etc., is influenced by the note height. The direction of the stem, the presence of slurs and of other possible note markers, the positioning of fingering, bowing, etc., depends on the position of the symbols in the neighborhoods. Relevant studies on music formatting processes are presented in (Roush, 1988), (Parrish, 1987), (Bellini, 2001), (Dannenberg, 1986), (Wolff, 1977).
This article presents the solutions adopted by the formatting engine of the Music editor called WEDELMUSIC (Bellini, 2003). Such object was developed in the homonymous research-oriented project co-founded by the European Commission. The WEDELMUSIC editor has been used for producing music sheets for large music orchestra performance, and it can be freely downloaded from the WEDELMUSIC WWW site. In addition, more recently, the Music Editor is in Open Source and other projects have already started exploiting its use.

The formatting engine for music notation presentation and sheet production is divided into two main systems: the insertion and positioning module and the justification and line-breaking module. The justification and line-breaking algorithms are not described in this paper, but for any further explanation, reference can be found in (Bellini, 2002).

The Music Intelligence Formatting Language (MILLA) has been defined and implemented to carry out the insertion and positioning of music notation symbols. It is a rule-based language and engine, which can be used to describe a series of rules and conditions interpreted in real time by an inferential engine, in order to apply formatting rules. The MILLA rules define formatting actions: they assign specific values to specific parameters related to the display and visualization of the music notation symbols. For example, a rule may state that the stem must present a certain length. The actions to be performed are activated by the conditions based on the music context of the symbol which is under evaluation. The conditions are expressed in terms of conditional sentences and define a music scenario. For example, a music scenario can be a note not belonging to a chord, not in polyphony and with a height comprised in a certain range. The verification of a music scenario defined by a conditional sentence leads to the application of a certain formatting rule to the symbol under evaluation. The MILLA formatting engine is invoked whenever inserting a symbol. If needed, the user can manually force the decisions using the user interface. The MILLA formatting engine is also invoked when the application window in the GUI is redrawn. The contexts for performing the MILLA engine invocation will be better explained in the next Section.

In (Bellini, 2001), an early and incomplete version of the MILLA language was presented. This paper reports an extended version of the MILLA language and the tools that can be used for developing and debugging the formatting rules. The major improvements of the MILLA, when comparing with previous version in (Bellini, 2001), deal with beams crossing staffs and tablatures, insertion of the debugger, and possibility of including clef and key signature changes into beams.

The paper is organized as follows. In Section 2, the general architecture of the WEDELMUSIC editor's automatic formatting engine is presented. In Section 3, the potentiality of the MILLA (Music Intelligence Formatting Language) is explained with several examples. The proposed examples show how activating suitable formatting rules on the basis of the content becomes possible. In Section 4, conclusions are drawn.

2 GENERAL ARCHITECTURE OF THE FORMATTING ENGINE

In Figure 1, the general architecture of formatting and justification engines is drawn. The functionalities of each component of the architecture are outlined as follows. The window of the music editor is both the source and the evidence of the formatting and justification engines. The editor window is used for showing music scores and by the user for sending commands to insert, delete symbols and for the general editing of the music score. The symbol and command selections are performed through the menu and toolbars of the user interface.

In WEDEL music editor, the music score is modeled by using an object-oriented model, including main score and part in the same model. The parts are used to build the main score. The additional information for the main score organization are contained in a separate file. Music notation symbols retain all the relationships among the notation symbols and music structure in general, without considering the values of graphic details regarding the music formatting on the screen (e.g., absolute or relative coordinates) (Bellini, 2001a).

When the MILLA engine for the automatic formatting is activated, the process of inserting a symbol in the score involves some computation of conditions and activates the corresponding rules. The conditions depend on the music context, for example the stem direction of notes may depend on the note height in the staff. In the object-oriented model, there are permanent parameters, therefore the value is computed only once at the very insertion of the symbol in the score. Furthermore, some dynamic parameters are computed every time the editor window is redrawn on the screen. This is very important when the music is visualized in the windows for a Computer View of the music score (in a resizable window and on different devices); whereas it is less important when the goal of the algorithm is to arrange the symbols for producing a music sheet. The presentation of music notation in Computer Views is a must for educational applications and for the computer fruition of music score in libraries. In fact, the visualization of the music page is often not acceptable due to the lack of resolution when the whole page is visualized and for the complexity of navigation when the visualization allows to analyze the needed measures.

In order to make the process of music score visualization shorter, the MILLA formatting engine has been conceived as made up of two parts:

- the ***Insertion Engine*** evaluates permanent parameters and it is invoked at the symbol insertion;
- the ***Positioning Engine*** is invoked every time the music editor window is redrawn (for example on resizing or on score scrolling). This allows to estimate the dynamic parameters related to symbols positioning.

The formatting engine estimates the context needed to assess the conditions, according to the music notation symbol involved. To perform the context evaluation, the MILLA engine makes queries to the object-oriented model of music. The context evaluation permits to identify the rules to be applied for setting the parameters of the symbol being inserted or positioned with the appropriate values. Each activated rule points out at the value to be set for the parameter under evaluation. For example, the stem height, the starting position of the slur, etc.

Conditions and rules are written in MILLA language and are included in a unique file loaded when the music editor is opened. The entire set of MILLA rules and conditions can be reloaded so as to let the possibility of changing rules and reapplying them to the same music score with no need of closing and re-opening the music editor. This is very important for the development and tuning of new rules as described below.

The music score is loaded and saved from the object-oriented model in symbolic format. WEDEL format is a XML compliant format for modeling music and it includes constructs for the description of integrated music objects including audio, symbolic music, images, documents, cataloguing information (Bellini, 2001a), see also www.wedelmusic.org for the public DTDs. The symbolic music section consists in a descriptive formal language used for saving the symbolic music scores. It does not include graphics details such as the coordinates of symbols. WEDEL format describes the scoring information, the musical notation symbols (for example notes, markers, clefs), their relationships (for example beams, brackets).

Parameters related to the music visualization are computed by the system in real time, on the basis of the MILLA Positioning conditions. Some formatting parameters of the music notation symbols (for example direction of the note stems) are saved in the file and expressed in terms of simple symbolic relationships (for example, up and down of the stem, above and below for expressions concerning the note they are referred to). This is useful in order to cope with exceptions and to save their value once they are computed on the basis of MILLA rules. The context evaluation and the estimation of positioning parameters are focussed on the other symbols visualized on the screen. Therefore, the visualization is strictly dependent on the positioning module of the formatting engine. This is invoked every time the window is resized -- e.g., on score scrolling and in general when the visualization changes.

The estimation of horizontal spaces among music symbols is performed by the justification and the line breaking modules (see Figure 1), (Bellini, 2002).

The MILLA and the justification engines perform their task independently from one another. In the WEDELMUSIC editor, the justification and the line breaking engines can be independently disabled to see the difference and to see a stable music score when the music in inserted. On the contrary, with most music editors providing a justification engine, such engine is always enabled, thus causing confusion to the user who has to follow the evolution of the music score on the page.

The parameters set by the justification engine are the spaces among the score symbols and they are not related to the formatting parameters set with MILLA rules, since the MILLA estimates only relative displacements with respect to the other symbols. When music symbols are horizontally spaced by the justification module and formatted by MILLA positioning engine, the line-breaking module is capable of arranging music in order to fill the page/view margins. The three modules set different parameters, thus contributing to the resulting visualization of the music score on the computer screen as well as on the print page.

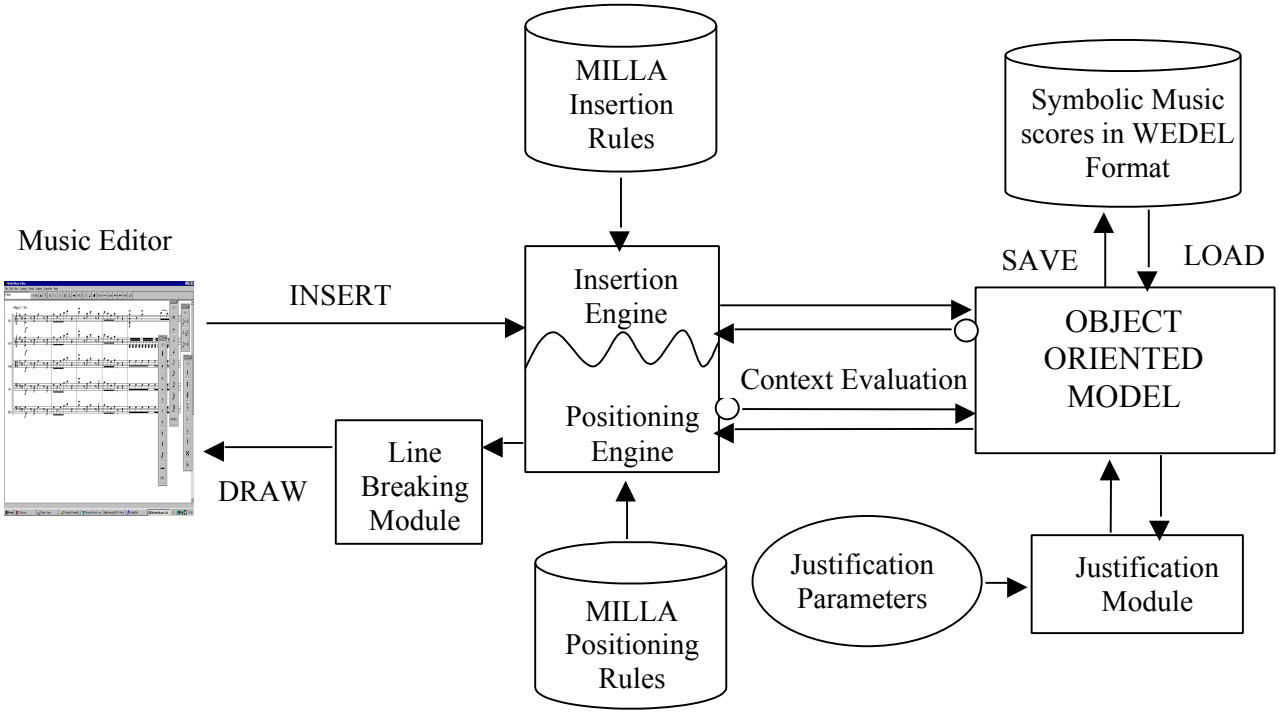


Figure 1 – general architecture of WEDEL Music Editor formatting engine

3 THE MILLA LANGUAGE AND APPROACH

Music formatting must address a large number of different music notation cases. In addition, the adopted solution allows to force some values in order to cope with the most relevant exceptions of music notation (some of them are described later on). The decision process of inserting and positioning music notation symbols in the music sheets must cope with a huge number of variables which characterize the music score context. This led us to abandon a strictly procedural organization of the formatting algorithms in favor of an inferential architecture for the formatting engine.

The MILLA formatting engine is a *rule based system*, where the rules stating formatting actions for music notation symbols are interpreted in real-time. The formatting actions define the values to be assigned to the parameters related to the displaying and the visualization of music notation symbols. The actions to be performed are activated by conditions based on the music context of the symbol under insertion and positioning.

For the evaluation of the music context a set of assertions can be described in order to take into account several aspects, such as: “is in polyphony”, or “note belonging to a chord”. These assertions can be true or false depending on the context of the symbol under evaluation. According to their values, different actions can be applied in different contexts. Single assertions can be combined with logical operators to specify more complex conditions to identify specific contexts where rules have to be applied. The AND connector is used to link together assertions in complex conditional sentences. This is a simple mechanism to insert/define specific rules for managing exceptions. They are always motivated and conceived as something to be activated by the specific context that brought about the exception or the typical formatting rule. The verification of a music scenario defined by a conditional sentence leads to the application of a given formatting rule for the symbol under evaluation.

The user can specify the conditional sentences by combining together the single assertions available and the rules to be applied in a certain context. This results in the ability of changing the formatting style of the score without spending too much time in extremely tedious activities like editing for modifying the music notation, symbols by symbol. This grants the Music Editor with a high flexibility, which relies on the flexibility of the MILLA language and formatting engine.

The set of rules and conditions can be planned and programmed in advance, but also changed at run-time to test/apply different layouts for a music score. The MILLA language is composed of words belonging to the music background and the syntax is quite simple so as to be intelligible for users without specific background on information technology. This allows to implement different music style sheets according to the publisher and end-users' needs and tastes. MILLA language is also bilingual, supporting both English and Italian.

The MILLA formatting engine is invoked at the very insertion of each symbol to set static formatting parameters. The rules activated during the insertion are the Insertion rules. If needed, the user can manually force the position through the user interface. In a second phase, the user can ask the system to evaluation once again the rules for reproducing the initial conditions.

The ***Insertion rules*** indicate the position of a symbol referred to another symbols. The graphic elements presenting values to be imposed by the Insertion formatting engine are reported in the following. For example:

- the stem direction of notes or chords, upward or downward;
- the stem direction for all the notes composing a beamed group;
- the relative position of markers (accents, expressions, ornaments, instrument symbols, bowing, fingering, etc.) with respect to the
  - note (up/down) they are referred to, either considering or not the direction of the note stem (on stem, opposite to stem).
  - staff (above/below), etc.;
- the automatic beaming of groups of notes depending on the time signature.

When redrawing the music editor window, for example for window resizing or for score page scrolling, the Positioning formatting engine evaluates the context to set dynamic parameters by applying Positioning rules. ***Positioning rules*** indicate the position of symbols with respect to the current visualization. This is expressed in terms of distance among other symbols in a GUI independent unit of measure (depending on the distance between two consecutive staff lines). This allows the correct visualization to be arranged regardless of the visualization magnitude.

The dynamic parameters estimated by Positioning rules are, for example:

- the stem length for notes, for chords and for notes and chords belonging to beams;
- the position and the angle/slope of the lines of beamed notes or chords;
- the coordinates of the markers with respect to the note they are referred to, expressed in terms of distance between staff lines (dy) and note head width (dx);
- the position of the symbols, inside or outside the staff, on the left or on the right of the notehead, etc.;
- the coordinates of horizontal symbols such as: slurs, bend, crescendo, decrescendo, change of octave, etc.

When more than one marker (expression, articulation, accent, etc., are generically called markers) is associated with a note, the ***Priority Rule*** is used. This rule specifies the order the positioning rules have to be applied with, in order to draw symbols around the figures (notes and rests are generically called figures). Symbols with higher priority are drawn closer to the figure even if they have been added to the score after lower priority symbols. The insertion of a high priority symbols implies the re-estimation of the position of the lower priority symbols. The priority rule has the form of a list, where symbols have a decreasing order of priority.

3.1 Syntax of rules and conditions

This section introduces the syntax of MILLA rules. In MILLA, there is a clear distinction between *rules*, which define the actions to be performed on parameters, and the *conditions*, specifying the context where the rules are activated.

The general form of conditions is:

<type>IF <*condition*> THEN <*rule name*>

Such conditions can be Insertion or Positioning and thus activating corresponding rules. Their distinguishing feature is the <type>, which can be INS or POS, for insertion and positioning, respectively. The <*condition*> is expressed as a conjunction of a set of assertions. For example, INMULTIVOICE and INCHORD, INSINGLELAYER, NUMUP-NUMD>0 and INCHORD, etc., these are three conditions. They permit to assess the music context as it will be explained hereafter. A condition being true activates the rules identified by the <*rule name*>.

The general form of a rule is:

<*type*> <*rule name*> *[DEFAULT]* <*symbol*> <*relationship*> <*value*>

Where:

- <*type*> identifies Insertion and Position rules and can be RULEINS or RULEPOS, respectively;
- <*rule name*> is used to associate the condition with the insertion rule;
- *DEFAULT* is a qualifier used to specify the highest priority rule among those activated by the same context. In addition, whether several conditions are true for the same symbol, the MILLA engine considers the condition providing the highest number of ANDed terms. When two conditions with the same number of terms are true in the same case (depending on the context), the rule with DEFAULT qualifier is applied;
- <*symbol*> is the reference symbol of the rule. The rule was meant to estimate a parameter for drawing and placing that symbol. For example: STEM, BEAM, TRILL, etc.;
- <*relationship*> states the type of relationship the rule is going to fix. For example, RELNOTE states that the relationship to be imposed on a given value is referred to the note for the mentioned <symbol>;
- <*value*> is a constraint stating the value to be imposed on the parameter identified by the combination of <symbol> <*relationship*>. For example, <symbol> STAFF ABOVE, <symbol> STAFF BELOW, STEM RELNOTE UP, STEM RELNOTE DOWN., which means defining that the STEM of the note has to be drawn on the upside part, when compared with the position of the note.

The syntax of the language presents INSIF, POSIF, RULEINS and RULEPOS instead of IF and RULE since the two sets of rules and conditions are loaded in different memory data structure, just to speed up their processing. In fact, as stated before, insertion rules are evaluated off-line while the position rules have to be processed in real time. This also allow to write insertion and position rules for related symbols in the same file and in the same section.

For example, an insertion rule “StemUp”, applied to symbol “STEM” of the note, which sets the stem upward with respect to the notehead can be stated as:

```
RULEINS StemUp STEM RELNOTE UP
```

A condition to activate this rule can be very simple. The condition could state that rule “StemUp” is applied if the note to be inserted is localized below the middle line of the staff (identified with a height equal to 0, see Figure 2):

```
insIF NOTE [VAR;0] THEN StemUp
```

The notation *[VAR;y]* means that the height of the note may range from the minimum to *y*, which means that is lower than *y*. Similar meanings have the notations *[y;VAR]* and *[x; y]*, respectively “higher than” and “comprised in the range”. Figure 2 shows the reference numbers assigned to the note heights referred to the lines and spaces of the staff in MILLA language. When tablatures are used the midline is always the 0.

STAFF

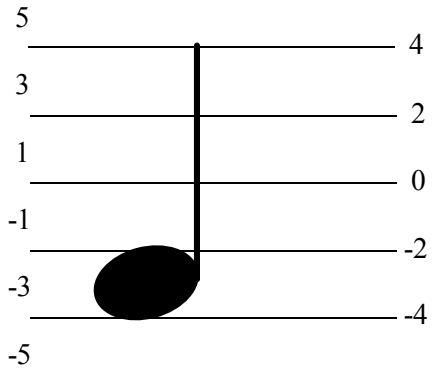


Figure 2 – the heights of the notes in MILLA with their position on the staff

Please note that, to have the positions numbered on the staff gives the possibility of defining the rules according to the position of the note on the staff. Other languages and models for music notation represent the position of the note on the basis of the note pitch and thus on the semantic interpretation of the note. The two models can be interchanged by taking into account the current clef and key signature.

A different condition may state that rule "StemUp" is invoked if the note belongs to the upper voice for a measure where polyphony (`INMULTIVOICEUPPER`) is present. The upper voice is the one presenting the note with the higher pitch among those to be played at the same time:

```
insIF NOTE INMULTIVOICEUPPER THEN StemUp
```

In the following paragraphs, an overview of some rules and conditions dealing with the parameters for some important symbols of the editor is presented. Please note that what is presented is not the full set of rules and conditions as it has been implemented and used in WEDELMUSIC Editor. The excerpts and examples of this paper have been produced by the WEDEL Music Editor and are proposed in order to give an idea of the MILLA approach.

3.2 Rules and conditions for note stems

These are the insertion and positioning rules, as well as the conditions for managing the position of the stems and their size.

- **Insertion Rules and Conditions for note stem direction**

The insertion rules for the notes specify the stem direction of a note or chord, which can be upward (UP) or downward (DOWN).

```
RULEINS StemUp STEM RELNOTE UP;
RULEINS StemDwn STEM RELNOTE DOWN;
```

These rules can be activated by the following conditions

```
insIF NOTE [0;VAR] THEN StemDwn;
insIF NOTE [VAR;-1] THEN StemUp;
```

The above two conditions represent the typical situation of note insertion. Condition NOTE [0;VAR] is true when the note highness is between 0 and infinity (VAR). If the note is localized in the middle line of staff or upper (`NOTE [0;VAR]`) rule "StemDwn" is called. For the opposite event (`NOTE [VAR;-1]`), rule "StemUp" is activated. In Figure 3, these two conditions are applied in the first measure.

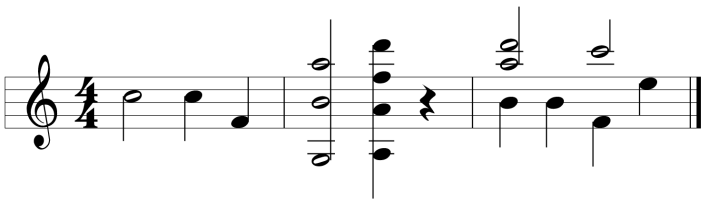


Figure 3 – the stem of notes and chords in single layer and in polyphony

In presence of more voices on the same staff, the following conditions can be applied:

```
insIF NOTE INMULTIVOICELOWER THEN StemDwn;
insIF NOTE INMULTIVOICEUPPER THEN StemUp;
```

Such conditions activate the appropriate rules depending on the voice the note has been inserted. The stem of the notes on the upper voice (`INMULTIVOICEUPPER`) must be upward, whereas the stem of the notes in the lower voice (`INMULTIVOICELOWER`) must be downward. In Figure 3, these conditions have been applied in the third measure for the single notes and also for the chords.
Another case is when the note is in a single layer and is included in a chord:

```
insIF NOTE INSINGLELAYER AND INCHORD AND UPESTD-LOWESTD>0 THEN StemDwn;
insIF NOTE INSINGLELAYER AND INCHORD AND UPESTD-LOWESTD<0 THEN StemUp;
```

Such conditions are met in the second measure of Figure 3. The notes belong to a chord (`INCHORD`), only one voice is present (`INSINGLELAYER`), and the difference between the highest and the lowest notes of the chord defines the "center of gravity" of the chord, either above or below the middle line, that is (`UPESTD-LOWESTD>0`). Where: UPESTD is the absolute value based on the distance between the highest note of the chord and the middle line of the staff, together with the values expressed in Figure 2. LOWESTD is the absolute value based on the distance between the lowest note of the chord and the middle line of the staff..

- **Positioning Rules and Conditions for note stem length**

The positioning rules of the stem affect its length. The basic unit for stem length is the *space* defined as the distance between two staff lines (Gourlay, 1987c). In this way, the standard length of the stem is 3.5 spaces, but it has to assume different values depending on the note height. In Figure 4, the following conditions have been used for some notes:

```
posIF NOTE[0;7] AND NOTE STEMDOWN THEN Stem3_5;
posIF NOTE[8;VAR] AND NOTE STEMDOWN THEN StemHeight;
```

The first condition is verified for the first note of the first measure of Figure 4, while the second conditions is true for the second note of second measure and for the first of the third measure.

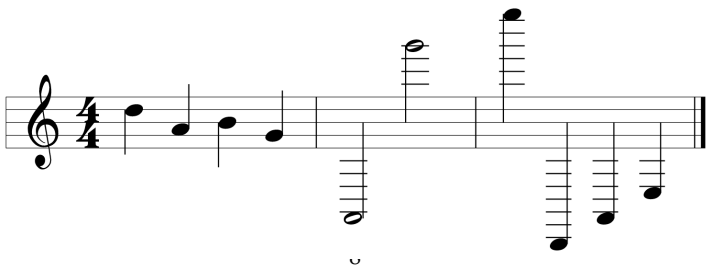


Figure 4 – the stem length for notes in single layer

The above-mentioned conditions represent only a small part of those needed to manage all possible scenarios. The positioning rules are defined in order to impose the stem length:

```
RULEPOS Stem2_5 STEM RELNOTE LENGTH=2.5;
…
RULEPOS Stem3_5 STEM RELNOTE LENGTH=3.5;
…
RULEPOS StemHeight STEM RELNOTE LENGTH=STEMHEIGHT;
```

The last rule is based on a special variable "STEMHEIGHT" that is dynamically estimated as the note height divided by two. This parametric rule is valid for all notes localized above the second additional staff line.

The following condition takes into account the case of polyphony and it occurs, for example, in the first note of the first measure of Figure 5 in the upper voice.

```
posIF NOTE[1;VAR] AND NOTE INMULTIVOICEUPPER THEN Stem2_5;
```

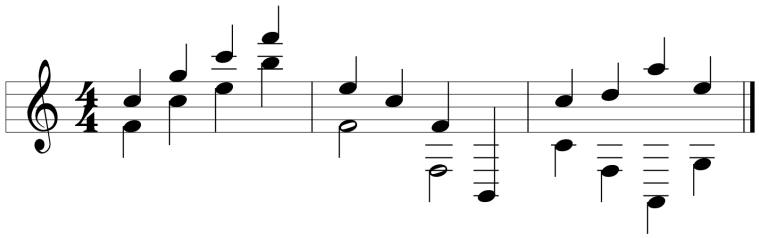


Figure 5 – the stem length for notes in polyphony

The above conditions activate one of the already presented rules. The others are not reported due to lack of space.

In the next table, the tokens that can be used with Notes and Chords are reported.

Affect symbol	Token Conditions	Token Rules	Actions
Note Chord	NOTE [x, y] NOTE INMULTIVOICEUPPER NOTE INMULTIVOICELOWER NOTE INMULTIVOICE NOTE INSINGLELAYER	STEMUP STEMDOWN	Set Stem direction: Up/Down
	INCHORD UPESTD – LOWESTD <*operator*> <*number*> NOTE STEMUP NOTE STEMDOWN NOTE2 NOTE1 NOTE1_2 NOTE1_4 NOTE1_8 NOTE1_16 NOTE1_32 NOTE1_64 NOTE1_128 STAFFLINES<*operator*> <*number*>	LENGTH= <*floating point number*>	Set Stem length

	STAFFSPAN<*operator*><*number*> ONSTAFFNUMBER=<*number*> NOTE HEAD=<*notehead type*> NOTEWITH <*marker symbol*>		

Where the Token Conditions have the following meaning:

- <*operator*> : < | > | = | >=| <=
- NOTE [x, y]: true when the note pitch is between the values indicated in the condition, the pitch is codified as previously explained: X, Y are two numbers defining the height range of the note; Example (NOTE [0, 3]);
- NOTE INMULTIVOICEUPPER: true when the note belongs to the upper voice, case of polyphony;
- NOTE INMULTIVOICELOWER: true when the note belongs to the lower voice, case of polyphony;
- NOTE INMULTIVOICE: true when the note belongs to a staff where there is more than a voice (polyphony);
- NOTE INSINGLELAYER: true when the note belongs to a staff where there is only a voice;
- INCHORD: true when the note belongs to a chord, or the rule is referred to the whole chord;
- UPESTD – LOWESTD upper and lower refer to the notes pitch,
  <*number*> : an integer number indicating the difference of the height of the highest and lowest notes of a beam. For example (UPESTD – LOWESTD = 0).
- NOTE STEMUP: true when the stem direction of the note is upward;
- NOTE STEMDOWN: true when the stem direction of the note is downward;
- NOTE2: true when the duration of the note is two measures. This permits to set different rules for notes having different duration and often different shape;
- NOTE1: true when the duration of the note is the whole measure;
- NOTE1_2: true when the duration of the note is 1/2;
- NOTE1_4: true when the duration of the note is 1/4 ;
- NOTE1_8: true when the duration of the note is 1/8 (croma);
- NOTE1_16: true when the note is a semicroma (duration 1/16);
- NOTE1_32: true when the note is a biscroma (duration 1/32);
- NOTE1_64: true when the note is a semibiscroma (duration 1/64);
- NOTE1_128: true when the note is a fusa (duration 1/128);
- STAFFLINES: the number of lines used for creating the generic staff. This permits to specify different rules for pentagram, tablature, percussion, etc., depending on the number of staff lines;
- STAFFSPAN=<*number*> in staff systems for piano, harp, organ, etc. which contain 2 or 3 staffs certain symbols can span across multiple staff as beams or chords, this qualifier allows to write conditions while distinguishing on how many staves the chord or the beam span. For example (STAFFSPAN=1 indicates a chord or beam spanning on only one staff);
- ONSTAFFNUMBER=<*number*> the staff systems for piano, harp, organ, etc. contain 2 or 3 staffs, this qualifier allows to write conditions while distinguishing in which staff the note is. For example (ONSTAFFNUMBER=0 indicates the upper staff of a multistaff part);
- NOTE HEAD=<*notehead type*> type of notehead chosen. This permits to define different rules for notes presenting different noteheads. For example, in presence of harmonics or when some guitar or percussion symbols are used; The currently supported noteheads are: CLASSIC, NOTE BREVE, CIRCLEX, X_HEAD, DDIESIS, DIAMOND, RHYTHMIC, TRIANG, TRIANG_UP, PLUS_HEAD, MOON, with a quite obvious meaning.
- NOTEWITH <*marker symbol*>: true when the note has the specified symbol around.
  Some example of value for <*marker symbol*> can be: STACCATO, SFORZATO, ACCENT, MARTELLATO, TENUTO, MARTDOLCE, PUNTOALLUNGATO, MORDENT, TURN, TRILL, WITHMUTE, WITHOUTMUTE, HARM, FINGER, TASTIERA, PONTICELLO, PUNTA, TALLONE, BOWUP, BOWDOWN, ARCO, PIZZICATO, STRING, etc..

3.3 Rules and conditions for beams on a single staff

Beamed notes are a group of notes having duration of 1/8 or shorter, which are drawn with a bar/beam grouping them together. For beams, it is very important to have flexible rules both for imposing the automatic beaming and estimating the slope of the beam. In the middle of a beam several other symbols can be found.

- **Insertion Rules and Conditions for beam orientation on a single staff**

Insertion rules for beamed notes affect the direction of the note stems belonging to the beam. All the stems of the beamed notes have to share the same direction (except for some specific cases, depending on the presence of multistaff parts, for further explanation see next sections). The context parameters to be evaluated in order to determine the stem direction of the beam depend on the note heights belonging to the group. For example, the group in the second measure of Figure 6 satisfies the following condition:

```
insIF STAFFSPAN=1 AND MEAN<0.0 AND NOTE INSINGLELAYER THEN BeamUp;
```

where STAFFSPAN=1 means that the beam has notes belonging to one staff, MEAN is the mean value of the note height:

$$MEAN = \frac{\sum note\ heights}{number\ of\ notes}$$

This parameter is useful to evaluate where the notes belonging to the beam are located in terms of pitch. Where:

- Σ note heights: summing the heights of all the notes of the group
- number of notes: number of notes in the group

In the 2nd and 4th measures below Figure 6 shows what happened after applying the following Insertion Rules for beaming,.

```
RULEINS BeamUp BEAM RELNOTE UP;
RULEINS BeamDwn BEAM RELNOTE DOWN;
```

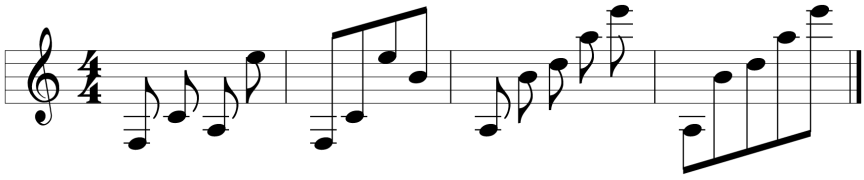


Figure 6 – the stem direction of a group of notes

- **Positioning Rules and Conditions for beam slope on a single staff**

Positioning rules for beams are used to impose their slope. It typically ranges from 0 (horizontal beam) to 0.5 (maximum slope corresponding to a 45° angle).

Figure 7 presents beams with different slopes. The slopes are typically decided considering the values of parameters such as the difference between the heights of the highest and lowest notes of the beam. While estimating the conditions for the identification of the right rule for beam slope definition, the value of the status variable BEAMDIS can be used as well. BEAMDIS is estimated as the distance in terms of height/spaces between the position of the highest and lowest notes:

$$BEAMDIS = |Highest| - |Lowest|$$

Where:

- Highest is the height of the highest note of the beam/group;

- Lowest is the height of the lowest note of the beam/group.

Heights are calculated assuming 0 for the midline of the staff/tablature. This parameter is useful to evaluate the dispersion of the pitch of the beam's notes around the central line of the staff/tablature.
For example, the following condition is typically used:

```
posIF STAFFSPAN=1 AND DELTA<0.5 THEN Beam0;
```

where DELTA is calculated by considering the number of notes composing the beam as:

$$DELTA = \frac{BEAMDIS}{Number\ of\ notes + 1}$$

This parameter is useful to evaluate the normalized dispersion of the pitch when considering the distance from the note with the highest pitch to the note with the lowest pitch and the number of notes belonging to the beam.

The above condition activates the following rule:

```
RULEPOS Beam0 BEAM RELNOTE ANYWHERE ANGLE=0.0;
```

Please note that the qualifier "RELNOTE ANYWHERE" is a <relationship> stating that the beam may cross the staff or can be located everywhere inside or outside the staff.

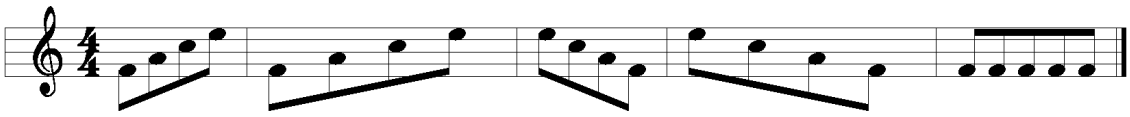


Figure 7 – some beams with different slopes

The Positioning rule can be used to set the beam slope assigning a variable value to the ANGLE parameter of the beam, furthermore the value can be constrained in a given interval:

```
RULEPOS Beam01_03 BEAM RELNOTE ANYWHERE ANGLE=[0.1,0.3];
```

The value of slope is calculated by dividing the difference between the Y coordinates of the first and last notes of the group and those on the X-axis. This value is estimated on the account of the decided justification parameter and line breaking.

The slope value obtained is constrained in the interval: if it is less than the minimum, the minimum slope is assumed and if the value is greater than the maximum, the maximum slope is assumed. However if the slope would lead to a note with a too short stem (less than 2 spaces for normal notes and less than 1 space for small notes) the horizontal beaming is used instead.

The condition allowing to take a decision about the rule for the beam slope setting can be defined by using the following value combined with MEAN:

$$MHL = \frac{Highest + Lowest}{2}$$

This parameter is useful to evaluate the mean value of the pitch of the notes considered

The beam slope is typically equal to 0 in presence of tablature or percussions:

```
posIF STAFFLINES<5.0 THEN Beam0;
posIF STAFFLINES>5.0 THEN Beam0;

RULEPOS Beam0 BEAM RELNOTES ANYWHERE ANGLE=0.0;
```

Affect symbol	Token Conditions	Token Rules	Actions
Beamed Notes	NOTES INSINGLELAYER NOTES INMULTIVOICEUPPER NOTES INMULTIVOICELOWER ONSTAFFNUMBER=<*number*> NOTE HEAD=<*notehead type*> NOTEWITH <*marker symbol*> STAFFLINES<*operator*> <*number*> MEAN <*operator*> <*floating point number*> BEAMDIS <*operator*> <*floating point number*> DELTA <*operator*> <*floating point number*> MEAN-MHL<*operator*> <*floating point number*> NOTE FIRSTLOWER NOTE FIRSTUPPER	STEMUP STEMDOWN ANYWHERE ANGLE <*operator*> <*floating point number*>	Set Stem direction of the group of notes: Up/Down Allow the beam to be placed anywhere Set slope of the beam

Some of the above qualifiers have been already described in previous tables. What follows is a comment only on the qualifiers which have been specifically introduced for this table.
The conditions on the beam refer to parameters extrapolated from the notes belonging to a beam.

- <*operator*> : < | > | = | >= | <= | ==
- MEAN: mean high value. For example: MEAN>1.0;
- BEAMDIS: beam dispersion. For example: BEAMDIS>0.0;
- DELTA: the value of the normalized beam dispersion. For example: DELTA>=0.0;
- MEAN-MHL: displacement with respect to the mean value. For example: MEAN-MHL <1.5;
- NOTE FIRSTLOWER: true when the first note of the beam is also the lower among the notes of the beam;
- NOTE FIRSTUPPER: true when the first note of the beam is also the upper among the notes of the beam.

3.4 Rules and conditions for beams on multiple staves

Multistaff music (scores for piano, organ, harp, etc.) needs more parameters to assess the context, with respect to single staff music scores. In case of multistaff parts it is possible to beam notes which are in different staves. To obtain the exact layout and slope desired, it is possible to write specific MILLA conditions. It is possible to choose where to place the beams: in case of three-staves parts you can place the beam above the staves, between the first and second staff, between the second and third staff or below the staves. There are MILLA rules and conditions for deciding both the position and the slope of the beam.

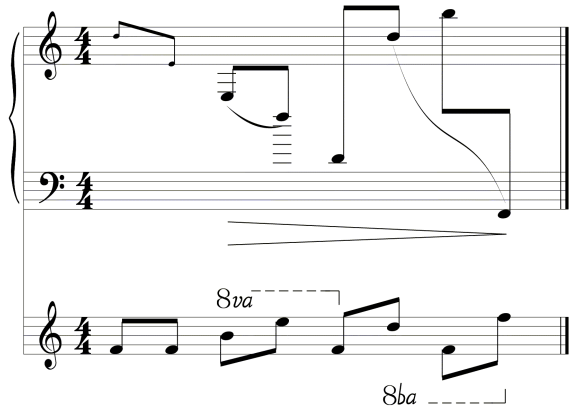


Figure 8 – A score with multistaff part

In order to allow the assessment of the context specific parameters related to the balance of the note heights belonging to the beam have to be provided. These parameters are similar to those introduced for the beam

rules in single staff parts. In effect, the parameters used are a generalisation in terms of meaning of the parameters for single staff parts presented before. When the beam spans more than one staff (STAFFSPAN>1) the height used in the calculation of the BEAMDIS, DELTA, MHL, MEAN parameters is transformed so as to allow the comparison of notes belonging to different staves. The height of a note is related to the staff where it is placed, this relative height is transformed to an absolute height using the formula:

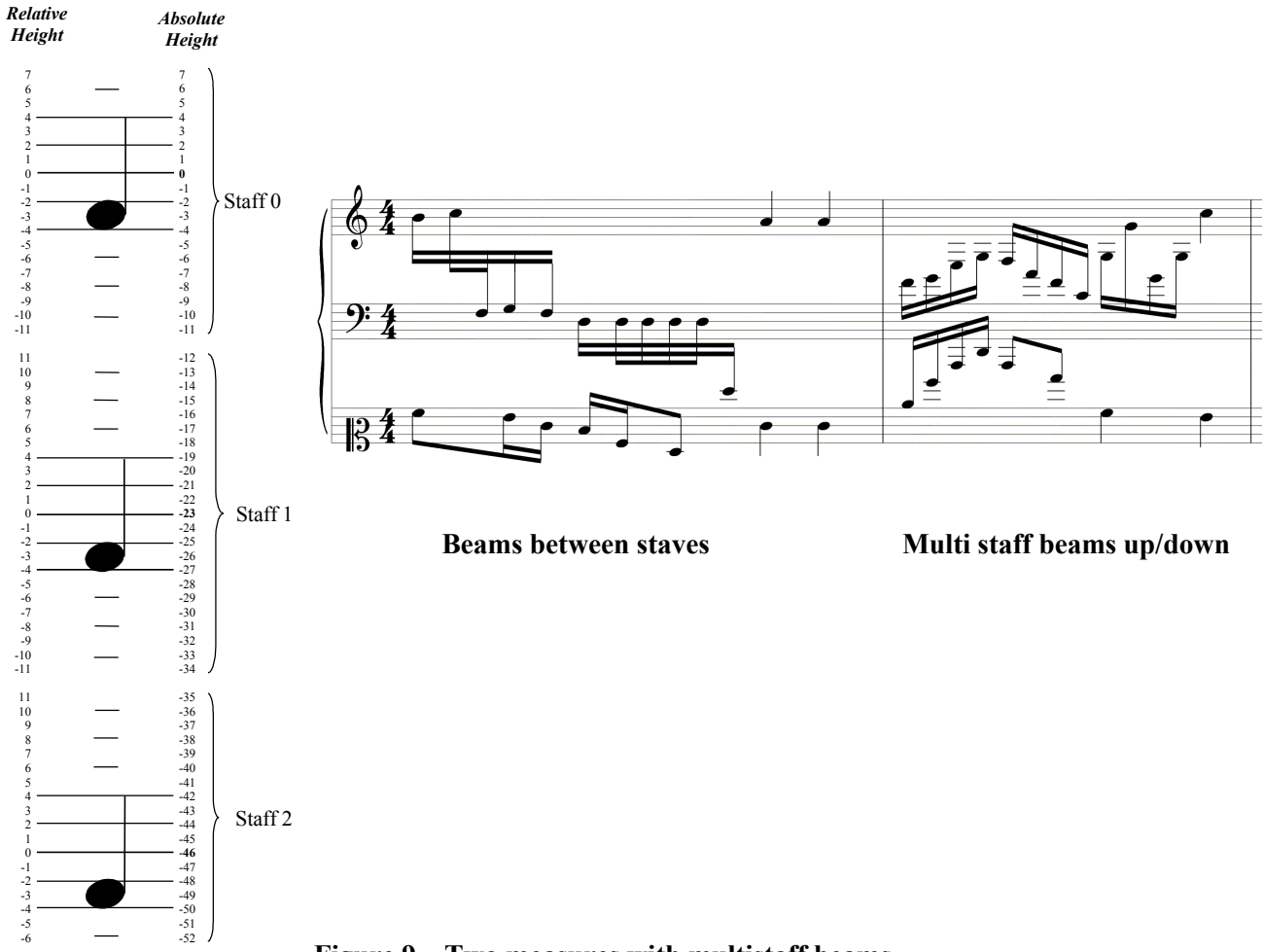


Figure 9 – Two measures with multistaff beams

$$AbsHeight = Height - 23 * Staff$$

where *Staff* is 0 for the upper staff, 1 for the mid staff or the lower staff in case of two staves, 2 for the lower staff in case of three staves. The mapping of the relative heights into absolute heights is depicted in the left-side part of Figure 9.

The choice of a fixed absolute scale which does not depend on the real graphic distance of the staves is due to the fact that, when using the graphic relations, the constants applied to constrain the parameters BEAMDIS, MEAN, etc. in the rule conditions would be valid for a certain staff distance but not for a greater or a minor one.

The constant 23 in the formula is arbitrary, and yet it comes out of a simple consideration, meaning that the usual distance of staves in piano scores is made of 8 spaces.

- **Insertion Rules and Conditions for beams in multistaff parts**

What follows is an example on how these parameters can be used in MILLA conditions:

```
RULEINS Beam12 DEFAULT BEAM RELNOTES BEAM12;
RULEINS Beam23 DEFAULT BEAM RELNOTES BEAM23;
```

…

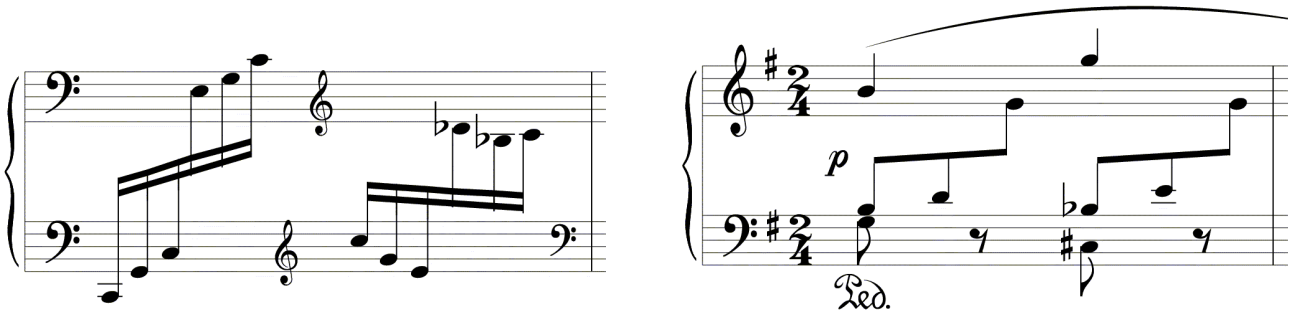
```
insIF STAFFSPAN>1 AND BEAMDIS<=-15 AND MEAN>=-23 THEN Beam12;
insIF STAFFSPAN>1 AND BEAMDIS<=-15 AND MEAN<-23 THEN Beam23;
insIF STAFFSPAN>1 AND BEAMDIS>-15 AND MEAN-MHL>=0 THEN BeamUp;
immIF STAFFSPAN>1 AND BEAMDIS>-15 AND MEAN-MHL<0 THEN BeamDwn;
```

…

- **Positioning Rules and Conditions for beams in multistaff parts**

What follows is an example on how the parameters defined above can be used in MILLA conditions:

```
RULEPOS BeamInt_015_03   BEAM RELNOTES ANYWHERE ANGLE=[0.15;0.3];
RULEPOS BeamInt_0_m007   BEAM RELNOTES ANYWHERE ANGLE=[0.0;-0.07];
RULEPOS BeamInt_0_007    BEAM RELNOTES ANYWHERE ANGLE=[0.0;0.07];
…
posIF STAFFSPAN>1 AND STAFFLINES=5 AND DELTA>-4 AND DELTA<-3 AND
BEAMDIS=-25 THEN BeamInt_015_03;
posIF STAFFSPAN>1 AND STAFFLINES=5 AND BEAMDIS=-22 THEN BeamInt_0_m007;
posIF STAFFSPAN>1 AND STAFFLINES=5 AND DELTA=-4 THEN BeamInt_0_007;
…
```



Different slopes for beams **Slopes almost horizontal**

Figure 10 – Two measures with multistaff beams

3.5 Rules and conditions for automatic beaming

Depending on the time signature of each measure, the MILLA engine automatically beams notes with hook/flag when a precise fraction of the time signature is reached during note insertion. For example, for the time signature of 4/4, the following condition can be imposed to activate rule "`NumBeat4`":

```
insIF TIME=4;4 THEN NumBeat4;
```

The referred insertion rule imposes the division of the measure in 4 beams:

```
RULEINS NumBeat4 MEASURE RELSTAFF NBEAT=4;
```

The system automatically beams a group of notes when a precise fraction of the time signature is reached.

Affect symbol	Token Conditions	Token Rules	Actions
Beamed Notes	TIME = num;den	MEASURE RELSTAFF NBEAT= <*number*>	Minimum number of notes the

		BEAMTUPLETS=OFF | ON	system tries to beam automatically during editing Automatic beaming of irregular groups

- MEASURE: stating that the condition and the rule are referred to the measure;
- TIME = Time Signature, expressed as num/den as in the first measure of the part;
- The values of *num*, *den* as above specified are the values of numerator and denominator of the time signature. For example: num=4, den=4;
- NBEAT= <*numbers*>, where the number has to be an integer value consistent with the time signature and stating the number of beat.

The information about time signature is very useful. The time signature of the measure can vary during the music piece. In addition, the time signature could express a real value different from what is specified in the staff. For example a time signature of 6/8 can be intended as two irregular groups for a total duration of 2/4. The time written on the score is 6/8, the real duration time is 2/4. This information is provided by the user through a dialog box. Information about time signature and real duration time is necessary to compute effective duration of the notes for visualization and midi output.

The rules for the automatic beaming set the parameter defining how many beats the measure should contain. In this context Beat means the number of groups the system tries to make inside the measure.

```
RULEPOS NumBeat1 MEASURE RELSTAFF NBEAT=1 BEAMTUPLETS=OFF;
RULEPOS NumBeat2 MEASURE RELSTAFF NBEAT=2 BEAMTUPLETS=OFF;
RULEPOS NumBeat3 MEASURE RELSTAFF NBEAT=3 BEAMTUPLETS=OFF;
RULEPOS NumBeat4 MEASURE RELSTAFF NBEAT=4 BEAMTUPLETS=OFF;
RULEPOS NumBeat5 MEASURE RELSTAFF NBEAT=5 BEAMTUPLETS=OFF;
…
posIF TIME=3;8 THEN NumBeat1;
posIF TIME=5;8 THEN NumBeat5;
posIF TIME=6;8 THEN NumBeat2;
posIF TIME=4;4 THEN NumBeat4;
…
posIF TIME=15;8 THEN NumBeat5;
posIF TIME=12;16 THEN NumBeat4;
…
```

3.6 Rules and conditions for markers

There are several symbols (in this context, called markers) that can be associated with the note. They are typically positioned above or below the note head. Among them:

- The expression symbols: staccato dot, sforzato, accento, martellato o staccatissimo, tenuto o sostenuto, martellato dolce, punto allungato, etc.
- The ornaments: turn, turn back, mordent, trill, tremolo, gliss, etc.
- The indication symbols: with mute, without mute, harmonics, fingering, etc.
- The instruments symbols: tastiera, bridge, punta, tallone, bowing, arc, pizzicato, string number, pedals, hell, toe, etc.

The expression symbols are listed below. In parenthesis the name of the related token in MILLA language :

- Staccato dot (STACCATO)
- Sforzato (SFORZATO)
- Accent (ACCENT)
- Martellato o Staccatissimo (MARTELLATO)
- Tenuto o Sostenuto (TENUTO)
- Martellato dolce (MARTDOLCE)
- Punto allungato (PUNTOALLUNGATO)

The ornament symbols are:

- Mordente (MORDENT)
- Gruppetto (TURN)
- Trill (TRILL)

The indication symbols are:

- In/Via (WITHMUTE, WITHOUTMUTE)
- Armonici (HARM)
- Fingering (FINGER)

The chord instruments symbols are:

- Keyboard (TASTIERA)
- Bridge (PONTICELLO)
- Punta (PUNTA)
- Tallone (TALLONE)
- Bow up (BOWUP)
- Bow down (BOWDOWN)
- Arc (ARCO)
- Pizzicato (PIZZICATO)
- String (STRING)

Some generic expression symbols can be defined by the user (see section 3.8).

All the symbols define a precise way of executing a note, so as to obtain a certain sound. In most cases, they are specifically defined for a given instrument. These symbols are always found in strict relationship with a note.

- **Insertion Rules and Conditions for Markers**

Insertion rules are used to estimate the position of markers. Their position is typically related to the direction of the note stem, if any. For example, in the next figure the insertion rule for the staccato-dot has been used by the first note of the first measure:

```
RULEINS Stac0 STAC RELNOTE OPPOSITETOSTEM;
```

It states that the staccato has to be drawn on the opposite side, with respect to the direction of the note stem. The opposite constraint is ONSTEM.

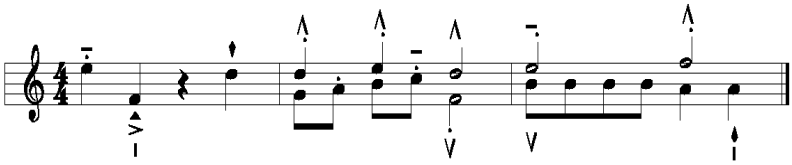


Figure 11 – Notes with markers

A condition activating the above rule is:

```
insIF NOTE INSINGLELAYER THEN Stac0;
```

- **Positioning Rules and Conditions for Markers**

Positioning rules of these symbols are used to impose parameters stating specific relative positions:

- in the space between staff lines or on the staff line: ONSPACE, ONLINE;
- outside or inside the staff: OUTSIDESTAFF, ONSTAFF;
- at an estimated distance from the note stem or from the note head: DX, DY;
- above or below the staff: ABOVE, BELOW.

A composition of such constraints helps to specify the exact position of the symbols with respect to the note and the staff. For example, for the staccato dot associated with the first note of the first measure as shown in the previous figure, the following rule was applied:

```
RULEPOS Stac2 STAC RELNOTE ONSPACE OUTSIDESTAFF DX=0 DY=0;
```

The rule was activated by condition:

```
posIF OPPOSITETOSTEM THEN Stac2;
```

The above conditions and rules represent examples showing how it becomespossible with MILLA to transform classical exceptions into rules.

Affect symbol	Token Conditions	Token Rules	Actions
	NOTE INSINGLELAYER	ONSTEM	Draws the symbol
STACCATO	NOTE INMULTIVOICE	OPPOSITETOSTEM	on the stem of the
SFORZATO	NOTE INMULTIVOICEUPPER		note or on the
ACCENT	NOTE INMULTIVOICELOWER		notehead
MARTELLATO	ONSTAFFNUMBER==<*number*>		
TENUTO	STAFFLINES==<*integer number*>		
MARTDOLCE	NOTEHEAD==<*notehead type*>	ONSPACE	States that the
PUNTOALLUNGATO	NOTEWITH <*marker symbol*>		symbol cannot be
	NOTE STEMUP		localized across a
Generic Accents	NOTE STEMDOWN		line
	NOTE [x, y]		
MORDENT		OUTSIDESTAFF	States that the
TURN	NOTE ABOVE		symbol must be
TRILL	NOTE BELOW		drawn outside the
	ONSTEM		staff
WITHMUTE	OPPOSITETOSTEM		
WITHOUTMUTE	TEN PRECEED STAC	ONSTAFF	States that the
HARM	STAC PRECEED TEN		symbol must be
FINGER			drawn inside the
			staff
TASTIERA			

PONTICELLO BOWUP BOWDOWN PUNTA TALLONE PIZZICATO ARCO STRING		DX=<*floating point number*> DY=<*floating point number*> ABOVE BELOW	Draws the symbol at the specified distance from the note (stem or notehead) Draws the symbol above or below the staff

Some of the above qualifiers have been already described in previous tables. What follows is a comment only on the qualifiers which have been specifically introduced for this table.

- NOTE ABOVE: true when the symbol is placed above the note;
- NOTE BELOW: true when the symbol is placed below the note;
- ONSTEM: true when the symbol is placed on the side of the note stem;
- OPPOSITETOSTEM: true when the symbol is placed on the side of the note head;
- TEN PRECEED STAC: true when the note the symbol refers to has the tenuto symbol closer to the notehead than to the staccato symbol;
- STAC PRECEED TEN: true when the note the symbol refers to has the staccato symbol closer to the notehead than to the tenuto symbol;

3.7 Priority Rule for symbols referred to notes

Markers and other symbols may refer to the same note. When this occurs, the order these symbols are drawn with is significant. The Priority Rule defines the order of positioning and accordingly symbols must be drawn above and/or below the note. The positioning does not depend on the order symbols have been inserted. The Priority Rule is applied when the editor window is redrawn.

To edit the priority rule is something which can be used to change the order of symbols without modifying the symbolic description of music score. The priority rule has the form of a list of symbol identifiers shown in decreasing order of priority. For example,

RULEPRI STACCATO TENUTO SLUR MARTELLATO ACCENT SFORZATO MARTDOLCE PUNTOALLUNGATO ARCO PUNTA TASTIERA BOWUP BOWDOWN STRING PIZZICATO,….., OCTAVE, …., etc.;

The above priority rule was applied in the following figure. The staccato-dot had a higher priority than the other symbols, as it can be seen in the first six notes. The PUNTA (a big P) marker has a higher priority than the PONTICELLO (`pont.`) marker and evidence is found in the first note of the third measure. The slur (SLUR) has less priority than the staccato-dot (STACCATO) and the tenuto marker (TENUTO), but a higher priority than any other symbol as shown by the slur between the second and third measure of the next figure..

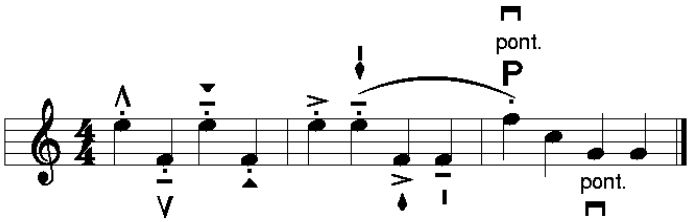


Figure 12 – The appearance order of the symbols is established in the priority rule

3.8 User-defined expression symbols

MILLA allows the definition of new symbols, which are considered as generic expression symbols related to a note. For these new symbols, insertion and positioning rules can be defined. Since symbols rules are usually very similar, symbols are grouped and rules are defined for group of symbols. However specific rules can be defined as well.

New groups can be defined using the DEFGROUP keyword:

```
DEFGROUP "faces";
DEFGROUP "stars";
…
```

A new symbol can be defined using the DEFSYM command:

```
DEFSYM "smile" GROUP="faces" FONT=”myexpr.fon” CODE=36;
```

specifying:

- the name of the symbol (“smile”),
- the group it belongs to (“faces”),
- the name of the font where such symbol can be found (“myexpr.fon”), if the font reference is omitted, then the name of the font is assumed to be the same as the name of the group,
- the code of the character representing the symbol (36).

The insertion and positioning rule for a group of symbols or for a specific symbol can be defined in the same way as for other symbols related to a note:

```
RULEINS faces0 DEFAULT "faces" RELNOTE OPPOSITETOSTEM;
RULEINS faces1 ALTERNATIVE "faces" RELNOTE ONSTEM;

insIF NOTE INSINGLELAYER THEN faces1;
insIF NOTE INMULTIVOICE THEN faces0;

RULEPOS faces2 "faces" RELNOTE ONSPACE OUTSIDESTAFF DX=0 DY=0;

posIF OPPOSITETOSTEM THEN faces2;
posIF ONSTEM THEN faces2;
…..
```

The symbol name (not the group name) has to be used in the priority rule to state the vertical relation of the symbol with other symbols:

RULEPRI STACCATO TENUTO SLUR MARTELLATO … PIZZICATO “smile” “star5” “sad” STRING …;

meaning that the “smile” symbol has to be over the pizzicato and below the “star5” symbol.

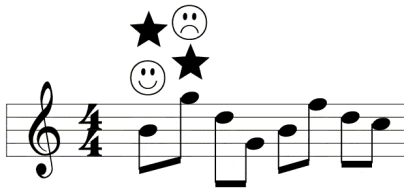


Figure 13 – Notes with user defined symbols

3.9 Note head rules

MILLA positioning rules can be defined to manage the note head appearance. Two kind of rules are related to the note head: (i) the rules used to find the proper font symbol to be used as note head, (ii) rules used to adjust the stem start position in relation with the note head type.

The note head can belong to one of the following types:

```
CLASSIC, DDIESIS, MOON, RHYTHMIC, TRIANG, TRIANG_UP, DIAMOND, PLUS_HEAD,
CIRCLEX, X_HEAD, ALPHANUM, ALPHANUM_SQUARE, ALPHANUM_REVERSE.
```

However, the note head type does not completely define the note head appearance (the font character code) and some other conditions have to be considered. For example the classic note head depends on the note duration: for note less than 1/2 the black note head is used, while for notes with duration greater or equal to 1/2 the white note head is used.

What follows is a list of some rules used for CLASSIC notes:

```
RULEPOS CodeEllipse HEAD RELNOTA CODE=216;
RULEPOS CodeMinima HEAD RELNOTA CODE=163;
…
posIF HEAD CLASSIC THEN CodeEllipse;
posIF HEAD CLASSIC AND NOTE MINIMA THEN CodeMinima;
…
```

Similar rules are used for rhythmic note heads:

```
RULEPOS CodeRhythmicT HEAD RELNOTA CODE=170;
RULEPOS CodeRhythmicB HEAD RELNOTA CODE=171;
RULEPOS CodeRhythmicW HEAD RELNOTA CODE=172;

posIF HEAD RHYTHMIC THEN CodeRhythmicT;
posIF HEAD RHYTHMIC AND NOTE SEMIMINIMA THEN CodeRhythmicB;
posIF HEAD RHYTHMIC AND NOTE MINIMA THEN CodeRhythmicW;
posIF HEAD RHYTHMIC AND NOTE SEMIBREVE THEN CodeRhythmicW;
```

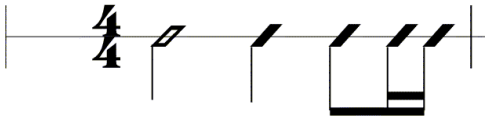


Figure 14 – Rhythmic note heads

Another positioning rule, which was defined with MILLA, is the stem start position. It depends mainly on the note head type and the direction of the stem (up/down).

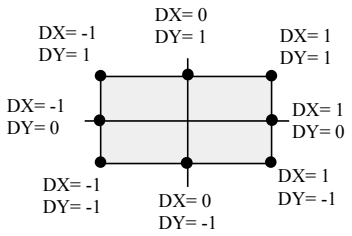


Figure 15 – The start stem position with respect to the note head

The start point of the stem is identified with respect to the note head center using DX and DY parameters that can be equal to –1, 0, 1 (left/bottom, center, right/up) subsequently 8 valid combinations are possible:

```
RULEPOS StemStartTopLeft STEMSTART RELNOTA DX=-1 DY=1;
RULEPOS StemStartMiddleLeft STEMSTART RELNOTA DX=-1 DY=0;
```

```
RULEPOS StemStartBottomLeft STEMSTART RELNOTA DX=-1 DY=-1;

RULEPOS StemStartTopCenter STEMSTART RELNOTA DX=0 DY=1;
RULEPOS StemStartBottomCenter STEMSTART RELNOTA DX=0 DY=-1;

RULEPOS StemStartTopRight STEMSTART RELNOTA DX=1 DY=1;
RULEPOS StemStartMiddleRight STEMSTART RELNOTA DX=1 DY=0;
RULEPOS StemStartBottomRight STEMSTART RELNOTA DX=1 DY=-1;

posIF HEAD CLASSIC AND NOTE STEMUP THEN StemStartMiddleRight;
posIF HEAD CLASSIC AND NOTE STEMDOWN THEN StemStartMiddleLeft;

posIF HEAD RHYTHMIC AND NOTE STEMUP THEN StemStartTopRight;
posIF HEAD RHYTHMIC AND NOTE STEMDOWN THEN StemStartBottomLeft;

posIF HEAD ALPHANUM AND NOTE STEMUP THEN StemStartTopCenter;
posIF HEAD ALPHANUM AND NOTE STEMDOWN THEN StemStartBottomCenter;
```

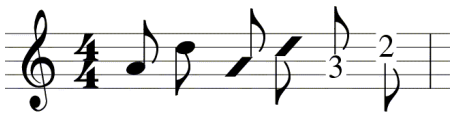


Figure 16 – Examples of stem start for classic, rhythmic and alphanum notehead

4 TOOLS TO OPERATE WITH MILLA LANGUAGE

4.1 Editing MILLA File

It is possible to edit and change from the WEDELMUSIC Editor the MILLA file. The MILLA file is a text file that can be edited with any text editor. It is possible to browse the MILLA file to change, add and delete the conditions and rules.

This will affect the scenarios where the rules are applied, and also the style of the score will undergo a change. Please note that the modified MILLA file is not automatically active until loaded again in the system using a specific command or by re-launching the WEDEL Editor.

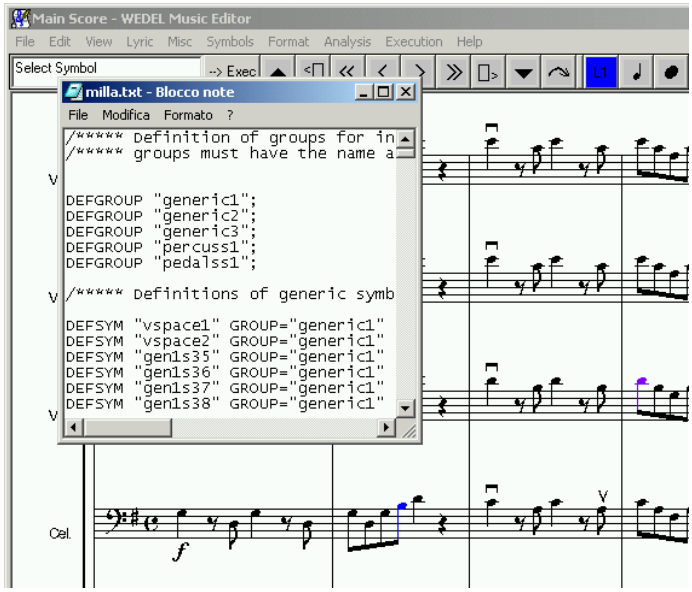


Figure 17 - Edit MILLA

4.2 Reloading MILLA file

After changing the MILLA file in order to define a new style of formatting, the user may want to see the changes in the score. To do this, the user must reload the set of rules and conditions in the music editor. Please note that only the parameters set by the Positioning Rules will be automatically re-estimated for the symbols already inserted in the current score. If any change occurred in Insertion rules and conditions, they would not be evident until any new symbol is inserted in the score. The reason for this is the following: the Insertion Rules set static parameters while the Positioning Rules set dynamic parameters. The Insertion Rules are applied only when the symbol is inserted for the first time in the score, while the Positioning Rules are estimated every time there is a change in the visualization window (for example a resizing). This separation produces a balance between performance (Positioning Rules take a lot of CPU time, Insertion Rules need less CPU time) and flexibility (Positioning Rules are more flexible and can be changed for the whole score, Insertion Rules are static and can be changed only for new symbols).

By using a specific command it is also possible to force the re-estimation of the Insertion Rules concerning the direction of the stem. This command is reported in the menu to allow the on-line trial of these rules without the reinsertion of the symbol.

4.3 Debugging MILLA Rules and Conditions

MILLA syntax is quite simple. In any case, a MILLA debugger is a very useful tool when it is necessary to change MILLA conditions or when a given exception has to be managed. For example, when a given exception is found, the MILLA code can be debugged in order to understand how this exception can be detected in terms of context condition. This allows to write the right detect condition and therefore to specify the behaviour of the system in certain scenarios, by adding tokens (simple statements) to the existing conditions.

The MILLA debugger is conceived as a dialog box, such dialog being associated with a different symbol: it provides help to the user because it presents all the available tokens in the system while specifying if those statements, when applying to current symbol, have already been verified or not. The debugger presents also the applied rules for the symbol and it specifies also which is the condition applied in the current scenario

(thus leading to the applied rule).

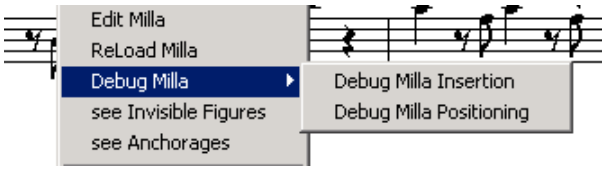


Figure 18 – Debug MILLA options in the Format Menu

The Insertion conditions and rules can be debugged only for newly inserted symbols. When setting the status of *Debug Milla Insertion* the system enters a state of wait. From that moment on, the newly inserted symbol will be debugged during its Insertion. For example, the insertion of a new note in the score will prompt the user to a Debug MILLA Insertion window, before any displaying of the score with the new note. The debug window presents all pieces of information about the scenario of the note and according to the latter the user may need to change the rule conditions in the MILLA file. In the left part of the dialog box, there is a list of all the available assertions which can be used to construct conditional sentences, in the right part of the dialog box (see below) there is a list of the conditional sentences already verified and defining the current scenario and the applied rule invoked by that condition.

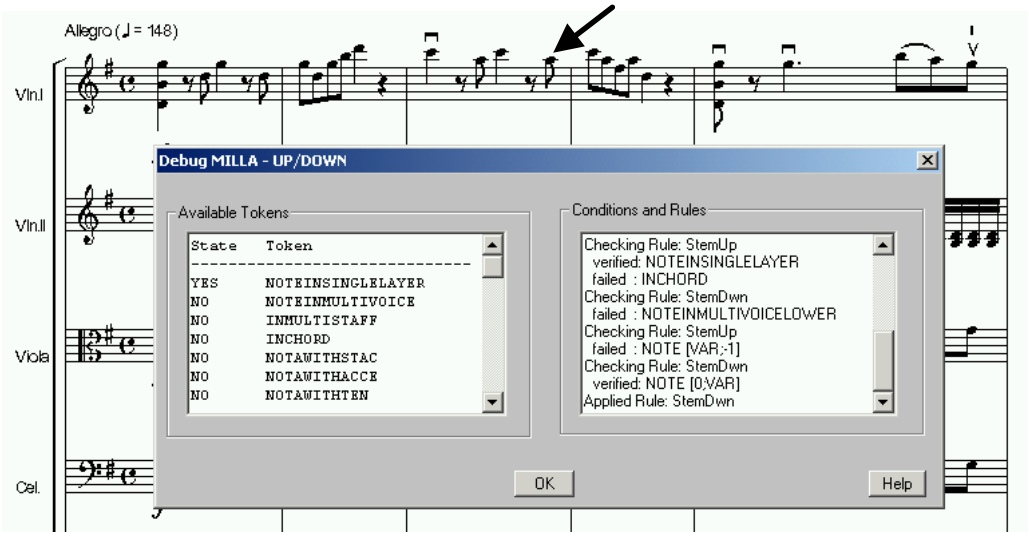


Figure 19 – Debug MILLA Insertion

To debug the Positioning rules and conditions, the user can set the state of the system in *Debug Milla Positioning* (see Figure 20). Now when the user selects any symbol of the score with the mouse, the system displays a dialog box. The information displayed in the dialog box is similar to the information shown in the dialog box of the Debug MILLA Insertion: there is a list of all the available conditions and their basic tokens used in the system to describe a music scenario, specifying also if such tokens are verified or not in the current context, and there is a list of the rules applied and of the conditions leading to the rule application.

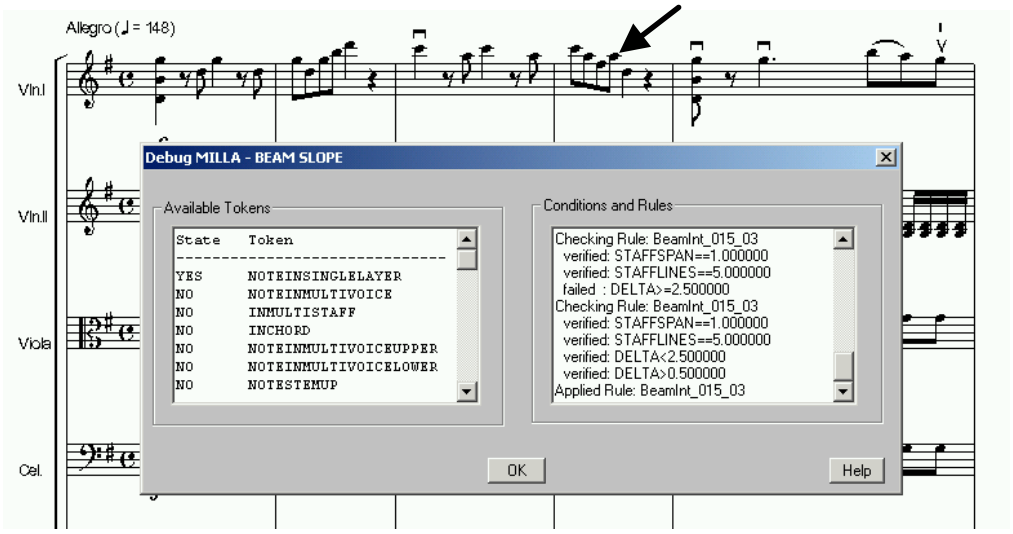


Figure 20 – Debug MILLA Positioning

For some parameters like the beam slope some mathematical parameters have to be evaluated: they are related to various controls to check the balance of the beam beginning with the height of the notes composing the beam.

Whenever a symbol is targeted by more rules, then more dialog boxes of Debug MILLA will open. Each dialog is oriented to address and debug related conditions and scenarios.

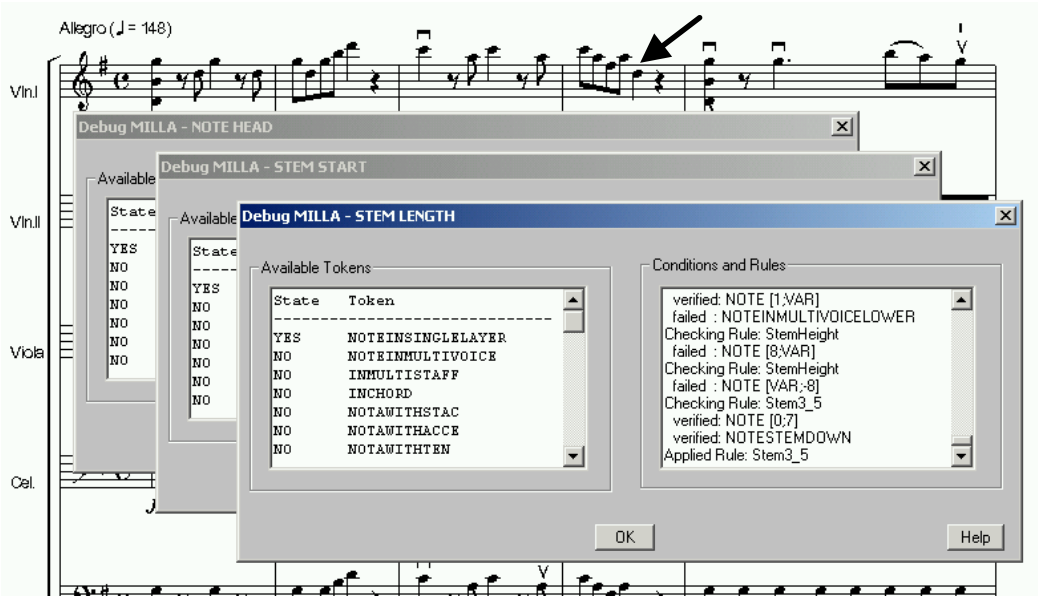


Figure 21 – Debug MILLA Positioning with two debug windows opened by the system

In the above example, by clicking on the highlighted note the system begins to debug the rule related to the note head (classic, x-head, sharp-head etc.), the rule related to the stem start position and the rule related to the stem length (which is the active window in Figure 21).

Figure 22 shows an example of rules tested to determine the beam slope. After being checked, rules are followed by the conditions verified or failed. The last is the applied rule that verifies all conditions.

```
Checking Rule: BeamInt_015_03
  verified: STAFFSPAN==1.000000
  verified: STAFFLINES==5.000000
  failed  : MEAN-MHL>1.500000
Checking Rule: BeamInt_015_03
  failed  : STAFFSPAN>1.000000
Checking Rule: BeamInt_0_015
  verified: STAFFNUMBER==1.000000
  verified: STAFFLINES==5.000000
  failed  : DELTA==4.000000
Checking Rule: Beam_0
  verified: STAFFSPAN==1.000000
  verified: STAFFLINES==5.000000
  failed  : MEAN-MHL<-1.500000
Checking Rule: BeamInt_03_035
  verified: STAFFSPAN==1.000000
  verified: STAFFLINES==5.000000
  failed  : DELTA==9.000000
Checking Rule: BeamInt_0_007
  verified: STAFFSPAN==1.000000
  verified: STAFFLINES==5.000000
  failed  : DELTA==2.000000
Checking Rule: BeamInt_015_03
  verified: STAFFSPAN==1.000000
  verified: STAFFLINES==5.000000
  failed  : MEAN-MHL>1.500000
Checking Rule: Beam_0
  verified: STAFFSPAN==1.000000
  verified: STAFFLINES==5.000000
  failed  : MEAN-MHL<-1.500000
Checking Rule: BeamInt_015_03
  verified: STAFFSPAN==1.000000
  verified: STAFFLINES==5.000000
  verified: DELTA>-2.500000
  failed  : DELTA<-0.500000
Checking Rule: BeamInt_015_03
  verified: STAFFSPAN==1.000000
  verified: STAFFLINES==5.000000
  failed  : DELTA>=2.500000
Checking Rule: BeamInt_015_03
  verified: STAFFSPAN==1.000000
  verified: STAFFLINES==5.000000
  verified: DELTA<2.500000
  verified: DELTA>0.500000
Applied Rule: BeamInt_015_03
```

Figure 22 – Log of tested rules for beam slope

5 CONCLUSIONS

Music formatting is an extremely complex task to be automatically performed, since it is based on the evaluation of a huge number of parameters. The solution proposed in this paper provides solution to many of the most common problems of music formatting. It allows to specify music formatting styles in terms of MILLA conditions and rules. A customized set of rules can be used to personalize the music formatting process. Therefore, users may change the music format by changing the music style without manually reformatting the music score symbol by symbol. At present the MILLA's rules and conditions deal with many aspects of music formatting in the so called Western music, but of also they can cope with guitar tablature, percussions, etc.

MILLA language can be enhanced by implementing new tokens (assertions on the music context) and writing new rules and conditions. MILLA has been used as the basis of the WEDELMUSIC formatting engine. The tool can be downloaded for free and the music editor is currently in Open Source.

6 REFERENCES

(Bellini, 1999) Bellini, P., Fioravanti, F., & Nesi, P. "Managing Music in Orchestras", IEEE Computer, Sept., 1999.

(Bellini, 2001) Bellini, P, Della Santa, R., Nesi, P. "Automatic Formatting of Music Sheets". Proc. of the First International Conference on WEB Delivering of Music, WEDELMUSIC-2001, IEEE Press, 23-24 November, Florence, Italy, pp. 170-177, 2001.

(Bellini, 2001a) Bellini, P., Nesi, P. "WEDELMUSIC FORMAT: An XML Music Notation Format for Emerging Applications", Proceedings of the 1st International Conference of Web Delivering of Music. 23-24 November, Florence, Italy, pp.79-86, IEEE press, 2001.

(Bellini, 2002) Bellini, P., Nesi, P. " Automatic Justification and Line Breaking of Music Sheets", Technical Report, Department of Systems and Informatics, University of Florence, UNIFI-DSI-TR-032-2002, 2002.

(Bellini, 2003) Bellini, P., Bethelemy, J., Bruno, I., Nesi, P., Spinu, M. B. "Multimedia Music Sharing among Mediateques, Archives and Distribution to their attendees", Journal on Applied Artificial Intelligence, Taylor and Francis, in press, 2003.

(Blostein, 1991) Blostein, D. and Haken, L. "Justification of Printed Music", Communications of the ACM, Vol.34, n.3, pp.88-99, March, 1991.

(Byrd, 1984) Byrd, Donald, A., "Music Notation by Computer", Indiana University, Dept. Of Computer

Science, Ph. D. Thesis, 1984.
(Dannenberg, 1986) Dannenberg, R., "A Structure for Representing, Displaying and Editing Music", In P. Berg (editor), Proceedings of the International Computer Music Conference 1986, pp.153-160. International Computer Music Association, 1986.
(Dannenberg, 1993) Dannenberg, R., "A Brief Survey of Music Representation Issues, Techniques, and Systems", published as: Dannenberg, "Music Representation Issues, Techniques, and Systems," Computer Music Journal, vol.17, n.3, pp.20-30., 1993.
(Gourlay, 1986) Gourlay, John S., "A Language for Music Printing", Communications of the ACM, Vol.29, May, pp.388-401, 1986.
(Gourlay, 1987) Gourlay, John S. and Parrish, Allen, "Computer Formatting of Musical Simultaneities," Ohio State University, Dept. Of Computer and Info. Science, OSU-CISRC-10/87-TR28, 1987.
(Gourlay, 1987b) Gourlay John S., Allen Parrish, Dean K. Roush, F. Javier Sola, and Yiling Tien, "Computer Formatting of Music," Technical Report, OSU-CISRC-2/87-TR3, Department of Computer and Information Science, The Ohio State University, 1987.
(Gourlay, 1987c) Gourlay, John S., "Spacing a Line of Music", Technical Report, OSU-CISRC-10/87-TR35, Department of Computer and Information Science, The Ohio State University, 1987.
(Hegazy, 1987) Hegazy, Wael A., and Gourlay, John S., "Optimal Line Breaking in Music," Ohio State University, Department Of Computer and Information Science, OSU-CISRC-10/87-TR33, 1987.
(Heussenstamm, 1987) Heussenstamm, G., "The Norton Manual of Music Notation", Ed. W. W. Norton & Company, Inc., 1987.
(Parrish, 1987) Allen Parrish, Wael A. Hegazy, John S. Gourlay, Dean K. Roush and F. Javier Sola, "Musicopy: An Automated Music Formatting System," Ohio State University, Dept. Of Computer and Info. Science, OSU-CISRC-10/87-TR29, 1987.
(Rader, 1996) Rader, Gary M., "Creating Printed Music Automatically", Computer, June 1996.
(Ross, 1970) Ross, Ted, "The Art of Music Engraving and Processing", Hansen Books, Miami, 1970.
(Roush, 1988) Roush D., "Music Formatting Guidelines" Ohio State University, Dept. Of Computer and Info. Science, OSU-CISRC-3/88-TR10, 1988.
(Wolff, 1977) A. Wolff, "Problems of Representation in Musical Computing", Computers and the Humanities, Vol.11, pp.3-12, 1977
(Wood, 1989) D. Wood, "Hemidemisemiquavers...and other such things. A concise guide to music notation", The Heritag Music Press, Dayton, Ohio, USA, 1989.